Filtering

/*There are many different ways to filter out unwanted data. You can look for specific values, sets of values, or ranges of values to include or exclude, or you can use various pattern-searching techniques to look for partial matches when dealing with string data.*/

/*The operators used within conditions include:

- Comparison operators, such as =, !=, <, >, <>, like, in, and between
- Arithmetic operators, such as +, –, *, and / */

-- Equality Conditions

```
SELECT
    customer.email
FROM
    customer
        INNER JOIN
    rental ON customer.customer_id = rental.customer_id
WHERE
    DATE(rental.rental_date) = '2005-06-14';
```

/*Inequality conditions

When building inequality conditions, you may choose to use either the != or <> operator.*/

```
SELECT
    customer.email
FROM
    customer
        INNER JOIN
    rental ON customer.customer_id = rental.customer_id
WHERE
    DATE(rental.rental_date) <> '2005-06-14';
```

/*Data modification using equality conditions

Since MySQL sessions are in auto-commit mode by default, you would not be able to roll back (undo) any changes made to the example data if one of my statements modified the data.*/

```
DELETE FROM rental
WHERE
    YEAR(rental_date) = 2004;

DELETE FROM rental
WHERE
    YEAR(rental_date) <> 2005
    AND YEAR(rental_date) <> 2006;
```

/*Range Conditions

Along with checking that an expression is equal to (or not equal to) another expression, you can build conditions that check whether an expression falls within a certain range.*/

```
SELECT
    customer_id, rental_date
FROM
    rental
WHERE
    rental_date < '2005-05-25';

SELECT
    customer_id, rental_date
FROM
    rental
WHERE
    rental_Date <= '2005-06-16'
        AND rental_date >= '2005-06-14';
```

/*The between operator

When you have both an upper and lower limit for your range, you may choose to use a single condition that utilizes the between operator rather than using two separate conditions*/

```
SELECT
    customer_id, rental_date
FROM
    rental
WHERE
    rental_date BETWEEN '2005-06-14' AND '2005-06-16';
```

/*When using the between operator, there are a couple of things to keep in mind. You should always specify the lower limit of the range first (after between) and the upper limit of the range second (after and).*/

```
desc payment;
```

```
SELECT
    customer_id, payment_date, amount
FROM
    payment
WHERE
    amount BETWEEN 10.0 AND 11.50;
```

/*String ranges

To work with string ranges, you need to know the order of the characters within your character set (the order in which the characters within a character set are sorted is called a collation).*/

```
SELECT
    first_name,last_name
FROM
    customer
WHERE
    last_name BETWEEN 'FA' AND 'FR';
```

```
SELECT
    first_name,last_name
FROM
    customer
WHERE
    last_name BETWEEN 'FA' AND 'FRB';
```

/*Membership Conditions*/

```
SELECT
    title, rating
FROM
    film
WHERE
    rating = 'G' OR rating = 'PG';
```

```
SELECT
    title, rating
FROM
    film
WHERE
    rating IN ('g' , 'pg');
```

/*Using Subqueries*/

```
SELECT
    title, rating
FROM
    film
WHERE
    rating IN (SELECT
            rating
        FROM
            film
        WHERE
            title LIKE '%PET%');
```

/*Using Not In*/

```
SELECT
    title, rating
FROM
    film
WHERE
    rating NOT IN ('G' , 'PG', 'PG-13');
```

/*Matching Conditions

So far, you have been introduced to conditions that identify an exact string, a range of strings, or a set of strings; the final condition type deals with partial string matches.*/

```
SELECT
    last_name, first_name
FROM
    customer
WHERE
    LEFT(last_name, 1) = 'Q';
```

/*Using wildcards

When searching for partial string matches, you might be interested in:

- Strings beginning/ending with a certain character
- Strings beginning/ending with a substring
- Strings containing a certain character anywhere within the string
- Strings containing a substring anywhere within the string
- Strings with a specific format, regardless of individual characters

```
_    Exactly one character
%    Any number of characters (including 0)*/
```

/*The search expression in the previous example specifies strings containing an A in the second position and a T in the fourth position, followed by any number of characters and ending in S.*/

```
SELECT
    first_name, last_name
FROM
    customer
WHERE
    last_name LIKE '_A_T%S';
```

/*Sample search expressions

```
F%                    Strings beginning with F
%t                    Strings ending with t
%pra%                 Strings containing the substring 'pra'
_ _ _ D E E P         Seven-character strings with a DEEP in the
last 4 characters
_ _ _-_ _-_ _ _ _     11-character strings with dashes in the
fourth and seventh positions*/


SELECT
    first_name, last_name
FROM
    customer
WHERE
    first_name LIKE 'M%'
        OR first_name LIKE 'P%';


SELECT
    last_name
FROM
    customer
WHERE
    last_name LIKE '%S'
        OR last_name LIKE '%N';
```

/*Using regular expressions

If you find that the wildcard characters don’t provide enough flexibility, you can use regular expressions to build search expressions. A regular expression is, in essence, a search expression on steroids. */

```
SELECT
    first_name, last_name
FROM
    customer
WHERE
    first_name REGEXP '^[PG]';


SELECT
    first_name, last_name
FROM
    customer
WHERE
    first_name REGEXP '^[NL]';


SELECT
    first_name, last_name
FROM
    customer
WHERE
    last_name REGEXP '^[ML]';
```

```
/* Null values

Not applicable:

Such as the employee ID column for a transaction that took place at an ATM machine

Value not yet known:

Such as when the federal ID is not known at the time a customer row is created

Value undefined:

Such as when an account is created for a product that has not yet been added to the database*/


desc rental;


-- You should not do it like this


SELECT

    customer_id, rental_id, return_date

FROM

    rental

WHERE

    return_date = NULL;


-- This is the way


SELECT

    rental_id, customer_id

FROM

    rental

WHERE

    return_date IS NULL;
```

```
SELECT
    rental_id, customer_id, return_date
FROM
    rental
WHERE
    return_date IS NOT NULL;
```

/*You have been asked to find all rentals that were not returned during May through August of 2005.*/

```
SELECT
    rental_id, customer_id, return_date
FROM
    rental
WHERE
    return_date NOT BETWEEN '2005-05-01' AND '2005-09-01';
```

/*While it is true that these 62 rentals were returned outside of the May to August window, if you look carefully at the data, you will see that all of the rows returned have a non-null return date. But what about the 183 rentals that were never returned? One might argue that these 183 rows were also not returned between May and August, so they should also be included in the result set. To answer the question correctly, therefore, you need to account for the possibility that some rows might contain a null in the return_date column*/

```
SELECT
    rental_id, customer_id, return_date
FROM
    rental
WHERE
    return_date IS NULL
        OR return_date NOT BETWEEN '2005-05-01' AND '2005-09-01';
```

EXERCISE:

/* You'll need to refer to the following subset of rows from the payment table for the first

two exercises:

payment_id	customer_id	amount	date(payment_date)
101	4	8.99	2005-08-18
102	4	1.99	2005-08-19
103	4	2.99	2005-08-20
104	4	6.99	2005-08-20
105	4	4.99	2005-08-21
106	4	2.99	2005-08-22
107	4	1.99	2005-08-23
108	5	0.99	2005-05-29
109	5	6.99	2005-05-31
110	5	1.99	2005-05-31
111	5	3.99	2005-06-15
112	5	2.99	2005-06-16
113	5	4.99	2005-06-17
114	5	2.99	2005-06-19
115	5	4.99	2005-06-20
116	5	4.99	2005-07-06
117	5	2.99	2005-07-08
118	5	4.99	2005-07-09
119	5	5.99	2005-07-09
120	5	1.99	2005-07-09

Exercise 1 - Which of the payment IDs would be returned by the following filter conditions?

customer_id <> 5 AND (amount > 8 OR date(payment_date) = '2005-08-23')

Ans - Payment ID - 101 and 107

Exercise 2 - Which of the payment IDs would be returned by the following filter conditions?

customer_id = 5 AND NOT (amount > 6 OR date(payment_date) = '2005-06-19')

Ans - Payment ID - 108,110 to 120

Exercise 3 - Construct a query that retrieves all rows from the payments table where the amount is either 1.98, 7.98, or 9.98.

*/

```
desc payment;


SELECT
    customer_id, amount
FROM
    payment
WHERE
    amount = 1.98 OR amount = 7.98
        OR amount = 9.98;


SELECT
    customer_id, amount
FROM
    payment
WHERE
    amount IN (1.98,7.98,9.98);
```

/*Exercise 4 - Construct a query that finds all customers whose last name contains an A in the second position and a W anywhere after the A.*/

```
SELECT
    last_name
FROM
    customer
WHERE
    last_name LIKE '_A%W%';
```